LIGHTING AND RENDERING

A guide through the Houdini help documents

Anybody got a good image to lend???

Docs designed for Houdini 11.0.547

INTRODUCTION

This document began inorder to help me learn more about lighting and rendering in houdini. It grew out of control from it's original inception as I added more and more info to it. At first it was me trying to make sense of the publicly available SESI help documentations, and then adding all the other information I learned over the years from trial and error and other software packages, plus the random stuff I pulled from the forums, especially odforce. I do believe this document will grow into a book by the end, and I believe the information will be helpful to everyone else. The more you know about you tools the better able you are to use them. Power drilling a screw driver, making a wedge of it, or chisleing with it may not be what was intended, but you can get more use out of it that way if you know you could use it that way.

You can think of this document as another resource to the many video tutorials that people teach especially Peter Quint, the two books The Magic of Houdini by Will Cunningham and Houdini on the Spot by Craig Zerouni, and all the classes and documentation that SideFX does. I understand the hard work they all went through so I give them a lot of credit. I went through most of this process on my own as I learned through these resources and I want as many people to be able to build off of what I did as possible, which is why I am posting this for free on odforce.

How to use this document

Ctrl+f or Command+f depending on your keyboard... I doubt many people are going to read this document as a page turner. I am trying to design this document as a book for ease of working, but in the end the info is more manageable if you take in chunks at a time as you work on a certain problem. If I leave info out on a parameter or a concept for instance the render engines, it is because I have already cited it previously, and want to avoid repeat information.

The flow of information will be based on covering the node interface from left to right through the tab menus. Certain important side topics may be contained within this node discussion. After the nodes are talked about I will insert the major sections of side topics on this info.

References to nodes, parameters, tab menu options, key board shortcuts, command line text, will be bolded to stand out in this document. There are many times a parameter name is similar to a larger concept for need of this, also I will try and keep the parameter names as to how they appear in the program their label names. Their technical names quite often vary from the label that is applied to the control so I'll try and make note of this, because of the importance of it.

Work in progress and Feedback

This document will always be a work in progress and I strongly encourage feedback. I am relatively learning as I write this stuff down. You will notice notes, clipped off information, and stuff not well organized at the end of the document. I am not an original prims user. I started in Houdini 9. Nor do I have a hundred years or VFX industry experience behind this book, at least until they learn how to clone people. So take this document as it is, as a work in progress, a selection of notes just really nicely organized, or a blog in book like form that fits more comfortably on your hard drive than your book shelf.

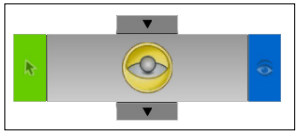
As for feedback spell check would be cool, but in reality info you've learned from the real world would be best, or if I've slightly got it wrong or completely got it wrong please tell me. I will fix it, eventually. You can apply back on this post or e-mail me through Ben@LaidlawFX.com. I try and have a day job and a life so the updates may be sporadic, if you harass me enough I can adjust my free time to update it, or you could potentially employee me.

Front page image by:

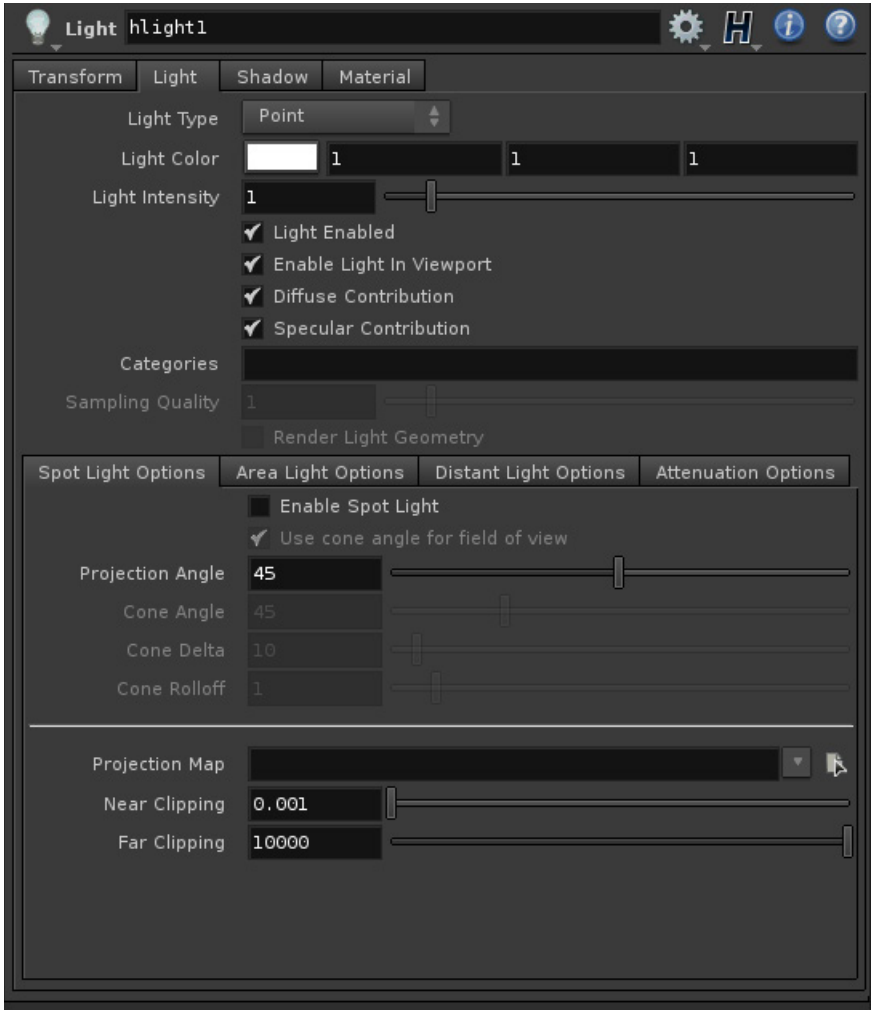
CONTENTS

INTRODUCTION TO THE LIGHT NODES

As with all thing in houdini lighting is done through nodes. These nodes are created at the object level of the scene. There are really only five light nodes and three of actual regular consequence. They are the light, the environment light, and the indirect light. The other two are the ambient and the light template while they can be used usefully are not as regurally used anymore and can be considered legacy.



In the tab menu and on the shelf there are more listed nodes, but these are different parameter presets of these five nodes. These include the point light, spot light, area light, geometry light, distant light, environment light, sky light, gi light, caustic light, and ambient light.



It is often useful to use the shelf to quickly set lights into the scene, since pressing tab will always place your light at the origin with default settings. There are a few short cut keys to be aware of the to help place the lights. LMB on the shelf will set your cursor to align a light on the X and Z axis based on the location of your cursor, pressing Alt subsequently will let you set Y axis height LMB in the viewport will final set your light at this location in the scene where Enter will set the object to the origin. If you press Ctrl prior to LMB on the shelf your light will be aligned to your viewport, be sure to check on the lock camera/light to view on the drop down menu in the viewport or on the stow tab on the right of the viewport. If you do by accident hit a node from the tab menu into the obj network or the shelf into the viewport just press delete after the node has been laid and it will delete that node. Further, take note of the tool tip help at the bottom of the screen for additional information or guidance.

Light

Point Light - default

Spot Light - Enable spot light is checked on

Area Light - Light type is set to grid

Geometry Light - Light type is set to geometry

Distant Light - Light type is set to distant

Environment Light

Environment Light - default

Sky Light - Drops two nodes; the Environment Light has enable sky environment map checked on, and the Light's light type is set to sun and has it's rotate and sun angle relative referenced to the rotate and sun angle of the sky environment map on the environment light

Indirect Light

GI Light - default

Caustic Light - Light type is set to caustic photon

LIGHT

The light node is the primary tool when you light your scene. It contains the majority of the features that would represent many other lights in several other packages, and each one of these options is a parameter so you can modify, then switch, and then mix match the parameters you may normally be accustomed to. This gives you great flexibiltiy with your lighting settups. If you just need a simple spot light at first then realize you need to upgrade it to an area light for soft shadows, and then need to add attenuation, and then shut off the spot light option you can very easily do so.

Transform tab

The transfrom tab is very similar to the geometry node parameters albeit it is just organized in a different manner. There are a few useful parameters to keep in mind when working with this node the keep position when parenting, icon scale, viewport selecting, and the look at option. If you are working with light rigs keep position when parenting is a thing to uncheck after moving your parent node around than will position your rig so you can adjust one light then reconnect it back to the rig. Under the viewing options tab Icon scale will change the size of your icon in the scene, where as normally the scale of the transform in other packages will do this in houdini in my most object nodes you have an icon scale. Also to take note, the icon display is the only thing shut off when you toggle off the display flag it does not stop the light from emiting light. Display will also only just make your icon appaear and disapear the slider has no control it is only an on off switch. Viewport selecting shuts off the ability to select the node in the viewport so you don't move it by accident. Select Scripts can set a script to run when the object is picked in the viewport like printing "Don't touch" or "This light is used for such and such".

Look at is one of the more useful function in lighting inorder to set a point to look at for an object. This look at point is best defined by a null object so that you can move it independently of the object your looking at. Most of the time the point that is mathmatically center on your object is not the artistic center of the object you want to define from the lights/cameras perspective. It is also a good option to animate the light by since you have full control of the point of focus as opposed to the controls in most software packages. You may want to RMB and lock parameter after you select an object so it does not break off. The animating along a path options are similar through out houdini just from an artistic stand point you may want to keep the path of your object separate fromt eh path of your camera. It will give you more creative freedom like using a null object for the look at object, you can allways set the path to a geometry object with an object merge sop or an edit sop to bend the points slightly from what they were.

Light tab

The first listed parameter is the Light Type this list, the point, line, grid, disk, sphere, geometry, distant, sun, and the legacy environment. There are four type of light emiting rays; point, directional, sun, and area. A point light emits light from one point in space in every direction. A direction light will cast parallel rays from one vector an infinite distance away. A sun light will cast rays from a finite source from an infinite distance away, it's best defined as a modified area light. A directional light is the simplest to calculate followd by a point light, a sun light, and an area light. Point and directional lights will both give you a sharp shadows. A sun light will give you a shadow falloff size that is controlled by the Sun Angle. An area light will cast the most natural soft shadow. The reason an area light will give you a soft shadow is that an area light has a surface area where there are two separate points in space that when they cast rays and hit the same tangent point on an illuminated object the distance between those two tangent is the falloff. The two farthest points on the lights surface that intersect on the same tangent spot defines the max falloff. Unlike a sun light this falloff control is affected by the distance the light is from the surface. In order to smooth out the shadow cast by an area light The more rays you cast via Sampling Quality the smoother this falloff will be. Increasing the Pixel Samples, the min-max samples, dicing, will further smooth this out as it runs the shader more often smoothinng out the impact points of the shaodw rays cast.

Inorder to create a point source with a shaodw falloff you have to write your own shadow shader and apply it to a Light template.

Standard Light settings

For the nodes Light, Environment Light, and Indirect Light there are a few basic parameters on all of the nodes.

- Light Color and Light Intensity - These set the color and intensity of the light. They are multiplied with each other.
- Light Enabled - The only switch to actually prevent the light from being rendered. It sets the intensity to zero and locks out the parameters on the light.
- Enable Light in Viewport - Shuts off light in viewport. This is good if the OGL can't handle the number of lights in the scene, or the light does not make sense to be displayed like an indirect caustic light.
- Diffuse and Specular Contributions - Enables the light to interact with these properties in the shader.

- Categories - As houdini develops look for this to replace bundling and masks.
- Render Light Geometry - This will render the geometry of an area light. This is good for identifying the actual brightness of an area light source when debugging the intensity of specular highlights, and to aid in the placement of light sources.
- Sampling Quality is for area lights. If you see noise created on the edge of the lights radius, it looks like shadow noise, and for noise in projection maps it will clean this up. It will increase the number of sample rays sent out from the light, this number is multiplied by the Pixel Samples in the mantra rop, under the properties, sampling tabs. When you are setting up for a shot you are going to develop, it is best to first adjust the Pixel Samples higher to your production standards, 3 x 3, 8 x 8, and 16 x 16 are common.

Spot Light Options

As mentioned previously spot lights are a modifier to point and area lights. To give the spot light parameters to any point or area light under the spot light tab check enable spot light. The light type of point plus spot lights will create the old fashion light icon. When not using a point light the icons will represent the standard light types. When a geometry light is used without transform into object checked the spot light parameter will define the edge of the light based on the profile of the geometry from the perspective of the light.

Cone angle will change the angle of projection while cone delta is often called penumbra, the fall off on the edge of the light. You can type in or MMB scroll to negative penumbra angles. Cone rolloff defines the mid point of penumbra fall off. One is the half point between the cone angle and cone delta. Zero to one pushes the midpoint in. Higher than one pushes the midpoint outward.

Projection map is not specific to spot lights you can use it with all light types, but it is easiest to use when defined by these parameters so it is under this tab. It projects a decal of your image although it may show as a repeating pattern in OGL. The projection area is determined by the vector, distance, and projection angle/cone angle of the light from the source object. For the point and area light the projection angle will resize the image, when the spot light is enabled the cone angle will resize the image, and when you uncheck use cone angle for field of view the image scale will revert to projection angle. For the distant and sun light orthographic width will resize the image. The legacy environment light type will project an image, but you should use the environment map instead. Using the projection map is a great way to find the size of your depth map size for your shadow map, since these are the controls for your shadow map size, will covere more on that later. Just use Mandril. rat for a quick image to test projection size.

Near clipping and far clipping will control the clipping plane of your projection map like the camera control on geometry. Using a test image you can MMB to scroll and see where a projected image will get cut off.

Area Light Options

Area lights as defined under the light types are line, grid, disc, sphere, geometry, and environment. As we mentioned sampling quality plays a significant role in image quality for these lights, and further you have more options under the Area Light Options tab. Area Size defines the X and Y size of the line, grid, and disc, while for the sphere the first channel defines X and Z. Without the Normalize Light Intensity to Area checked changing the area size of the light will increase the intensity of the light by a multiplication factor of the size of the new area size, but it will also disipate since the rays of light are coming from a larger area. To test this shut off specular contributioon, keep your intesnity at 1, and you'll notice the average values increase in mplay with the, i key, 1x1=.1, 2x1=.2, and 2x2=.4. When the normalize function is on you will allways be hitting around 1 no matter the area size of the light.

Single sided and reverse emission direction will change the direction of the emitted light. The best way to view this change in direction is to use a geometry light and give it normals with a point sop. The directions of the normals will be the forward direction of the light, while when you check reverse emission direction it will admit to the negative normal direction. For the line, grid, and disc the direction is on the positive and negative object Z-axis. For the sphere it is an external and internal direction as the normals would be on a polygons sphere.

The Environment map was originally designed for the legacy environment light type on the light node. Using the environment light node will give you more options now. This light It will tint your illumination based on the color from a given direction. For instance if you have a disc shaped light and it moves across the horizon it will take on the color of the sky. You can direct the map with the environment space tab, or use the environment null like the look at option in the transform tab.

Texture map will be multiplies by the color of the light based on the UVs of the geometry, it prodcues a similar result to the projection map. Enable edge falloff unlocks edge width and edge rolloff, which look similar to the penumbra controls on the image, as edge width will fade the image in by that fraction and rolloff controls the midpoint of that falloff.

Geometry Light

When the geometry light type is enabled you can use the geometry object parameter. This allows you to model any type of light source you want. By adding the Cd attribute to the sop level geometry you can alter the light content too since it is multiplied to the intensity. If you check transform into this object the light will ignore it's own object level transform and position itself at the center of the object space of the geometry node. This should be not confused with the fact that the path leads to the obj level of the geometry. So you can place the world position at the object level on the left side of the screen and go into the sop level and move the object position to the right side of the screen and that is where the light will emit from. This is because the light emits from the surface normals of the geometry.

If your light is not emitting the way you want or think it should be display the face normals, and if they are not pointing in the right direction you can tweak it manually with a point sop, or you can use insert a reverse sop to flip the face directions.

To better handle and optimize your geometry lights you should follow a few guides. Make sure your light emiting geometry is not renderable so you do not have shadow artifacts. Use the lowest poly geometry you can to make the shape of your light source. The higher the resolution the geometry, or geometry that is rendered as sub-divisions or nurbs will take longer to render and may cause alot more noise. Use open surface, single sided geometry, rather than closed meshes, a plane versus a box. Also delete geometry that does not contribute to the scene, mantra may perform lighting computations for hidden faces, so if it is not there it will not contribute.

Distant Light Options

Orthographic width and sun angle will adjust the width of your shadow maps on your distance based lights and control the amount of falloff on the shadow of the sun light type. After you set your light vector for the distant or sun light type it is good then to position your light for the best depth map projection so you should set your orthographic width then inorder to get the best coverage.

Attenuation Options

This is the intensity control over distance. It is multiplied with your light intensity and light color. There are three types of attenuations half distance, physically correct, and the ramp multiplier.

Half distance attenuation enables the half distance parameter which lets you set the half life/intensity positon of your light. So if your intensity is 1 and your half distance is 10, then at 10 units away from your light your intensity will be .5. Mathematically this is CI = H / (H + D), H is the half Distance, D is the distance from the light. It is important to note that the light is never really extinguished it technically goes to infinity. Using the ramp multiplier can cut this, or the active radius.

Physically correct is how light falls off in reality based on the inverse square law, CI = 1 / (D * D), this is a good setting for area lights since they are normally placed in a natural real world location to the object being illuminated.

Attenuation start which is used for both physically correct and half distance is used for the point and line light types. The light illumination from the light to this distance will be constant on diffuse surfaces within this distance.

The ramp multiplier can either work on it's on, or it will be mutliplied with physically correct or half distance attenuation. The end and begining of the ramp are clamped to the ramp start distance and ramp end distance. So if you make the end distance 100 your center point will be 50 on the ramp, and at 10 your center point on the ramp will be 5. This gives you creative control in the viewport, and it allows you to create any type of artistic falloff you want, whether it is a pulsing light, or a rainbow, since, the ramp multiplier, or any attenuation, is just intensity over distance.

Active radius is a good optimizing switch that will limit the distance at which the light is entered into light and shadow calculations. A good trick to do is to channel link to your ramp end distance or

a multiple of your half distance to shut your light off at a distance you know the light will not be doing any significant contribution.

Shadow tab

There are three types of shadows that can be created in Houdini. Under shadow type there are ray-traced shadows and depth map shadows. The third deep shadow maps is engaged when depth map shadows and transparent shadows is on. Transparent shadows is a switch that enables opacity calculations from the surface shader. So if you want the shadow to go beyond one layer of geometry you need to turn this on, very important for sprite and volume rendering. Raytracing lends to more realistic shadows on hero object that are close to camera and need to go from focused to blur where as either shadow map will blur evenly across the shadow not realistically, but are much cheaper to render since they can be look ups instead of created at render time.

Shadow mask allow you to select the objects to be shaded, * is a wildcard that select all objects, any object selected in the shadow linker pane will show here. Shadow intensity will manually dim or intensify the shadow. Clicking phantom on a geometry object will make the object cast shadow, but not be scene in the render.

The settings for depth map shadows and deep shadow maps are pretty similar the only two specifically for deep shadow maps are the pixel samples(under the shadow tab) and depth map motion blur. The former will increase the samples of the shadow at each pixel, while the later enables blur to be captured in the shadow map(you can use regular depth map shadows, but do to the quality it is not advisable). The two shadow maps store different information. Deep shadow maps are 3-D storage of the opacity of any translucent object the light ray passes through upto and including the depth of the final opaque surface. Each hit of a ray stores two channels Pz(depth) and Of(opacity) in multiple layers until it finishes. While the depth shadow map just stores the distance of a light to a surface, one channel Pz, stored in one layer so it is a simple image. Subsequently, deep shadow maps take up a lot more disk space than depth shadow maps.

To adjust the quality of shadow maps there are a few settings;

- Shadow bias this is a small distance that if occluders are within they will not be considered shadowers. This helps cleans up self-shadowing artifacts. If your scene is large increase the bias value to 5-10% to start.
- Shadow quality improves the quality of the lookups evaluating standard shadow maps, or blurred shadows.
- Shadow softness is a blur defined by the relative size of a micropolygon when rendering in micropolygon mode, but should be considered about a pixel otherwise.
- Shadow blur is a fraction of the image to be blurred. This will give you larger more controlled blur than shadow softness since it does not take into consideration surface derivates like shadow softness.
- Shadow maps if you are happy with your shadow maps make sure to shut off auto-generate shadow maps. Always use the .rat format to record shadows whether depth or deep it is more efficent than using .pic, and has better anti-aliasing. It is not advisable to use other outside formatts for shadow maps.
- Resolution is the primary quality control on shadow maps, the larger the map over the smaller the surface area the better the quality.

Material tab

This allows you to overide the area lights default shader, the “VEX Area Light”. So you insert that shader in there, or you can place a regular surface shader in there, but you must verify that it does not compute illumination itself.

Reference Documents: Light object, Lighting Shadows, Rendering Deep shadow maps and Deepcamera maps, light linking and shadow linking, mantra rop.

LIGHT RAYS AND FOG-VOLUMETRIC RENDERING

Light rays, also known as god rays, are robust, simple to settup, and quick to render with multiple cores. We are going to explain one simple method using the atmosphere node and a fog shader to achieve this lighting effect. This method entails volumetric rendering similar to smoke, clouds, spray, fire, and fluids which are all subjects of themselves, we will only cover setting up light fog and the basics of rendering fog-volumetrics as it is applied. It is important to note that a fog shader is a post-process of volumetric rendering and even though it does the same process as volumetrics rendering it composites over the surfaces, so semi-transparent surfaces don't composite the fog properly. Where as, rendering volumes and surfaces directly as micro-volumes with volumetric rendering will composite them correctly.

You can use this method of using the atmosphere node and a fog shader with all of the light types from the Light node except the legacy environment type. The point, distant, and sun type work the best, where as for the area lights you need to increase the sampling extremely high. It is best to set up your lighting first then add a few point based spot lights to add the light rays. If you don't see fog being produced you may need to either move your lights or increase the density of your fog, to test this process follow the simple scene settup below.

1. Create a grid and a sphere
2. Ctrl+LMB a spot light from the shelf and LMB Lock camera/light to view and position it so the cone of the light fits within the grid. Unlock the camera view and position it to frame the gird and the sphere.
3. Add a shopnet inside of which add a Lit Fog "Vex Light Fog" and set the fog density to .1
4. Add a atmosphere and set the material to v_litfog1 from your shopnet.

The two important nodes of this concept are the atmosphere node and the lit fog. The atmosphere node, labeled fog when you place it down, can really be just paired down to the material and light mask parameters. It is important to set the light mask parameter on this node like caustics so that you do not process lights, that you do not need, to save render time. The material parameter just links to your fog shader.

Lit fog and uniform fog are the most basic and straight forward prebuilt fog shader to use and are the best recommended. Uniform fog is just a constant layer of fog while lit fog allows you to apply a noise function to make the fog move which is best for a little added realism. Additional for the more advanced user you can use 3d texture fog, fog shader builder, pathtracer, photon tracer, z-depth fog, and z-map fog. These fog shaders are pretty straight forward based from their names, with the 3d texture fog, z-depth fog, and z-map fog being the most artistic and render decreasing methods, respectively.

Vops

It is important to note Houdini does ray marching to create these fog shader. Ray marching is a process in which shading points are generated in the volume by uniformly stepping along rays for each pixel in an image. For a simple thought of this, think of an army of soldiers marching down a road, for each soldier there is a sample of the fog and the road is the path of light. In Houdini this process is very similar to micropolygon rendering which we describe a lot more in depth later, but with depth in mind so it can be considered micro-volume rendering. Because of this process many of the same rendering process that apply to surface shading can be applied to micro-volume rendering, such as, color, opacity culling, instancing, shadow maps, depth of field, motion blur, and displacement. Displacement shaders in fact can move the micro-volumes, which can add detail to the volume cheaply.

There are a few things to be aware of though when writing a surface shader or displacement shader to work with volumes instead of surfaces. For surface shaders, you'll generally want to scale the Of output to take into account the volume step being shaded (dPdz). For displacement shaders, displacing along the normal displaces along the volume gradient, but what you'll usually want is to perform a 3D displacement that is independent of the normal. The density variable represents the volume density (passed to shaders as a float), and the global variable N represents the gradient vector (the direction of greatest change in the density) at the micro-volume being rendered.

On the fog shader themselves which controls fog as opposed to all of of volume rendering controls, which are at a more important inheritance level, are the step size and maximum steps. Decreasing step size and increasing the maximum steps improves quality while increasing redner time. Do shadows will also increase render times, but is on by default, since it is more natural to render with shadows.

Custom Volumetric Shaders

A good shader to break down if you want to create your own custom volumetric shader is the basic smoke. What defines these shaders from your basic shading vops is about fourteen nodes, and two flows of information. These two flows of information define the surface color and shadow as one flow, and the second flow is for the alpha and opacity which defines the shape and depth of the volumetrics. The surface color and shadow is the most simple. It is defined by an illuminace loop vop appended to the output variables an paremeters vop through the surface color. By default this will not cast shadow, so inside the node you should create a global variable and check output a single variable with variable name Light Color, which is only available inside the illuminance loop. Append a shadow vop and connect it into a next input into the sub output. The shadow vop will do the work inside the illuminace if connected by itself with settings to 1,1,1, but it is best to let the global variable define this so you don't have to match your light color to to that of your shaders light color, but it is good info if you want to break it.

For the alpha and opacity flow you need a series of nodes that begin with creating one parameter vop that's name is density, this calls a global variable that is not listed within the vop of a similar name. Also create a constant to multiply with the density value to control its intensity. An expected value for this density control is a float value of 50 for the test scene, this can be scene dependent so once the rest of the nodes are plugged in you can adjust this value. Append a maximum vop and add a constant float into the next input that defines the low value of 0, you can call this constant zero for ease. This sets a lower limit for the density values when they travel below zero. Append a negate to the maximum output to prepare it to be multiplied by the global variable change in position with depth, this defines the shape of the volume. To bring these values back into a working color space we append a exponential and complement vop. From here you can input this value into the alpha and opacity of the output variables and parameters vop, and you should also premultiply this by the output of the illuminace loop to define the shape of the surface color.

Additional volumetric rendering info

Contrary to what was stated before we are going to list some additional volumetric rendering information that may be helpful in rendering fog, but is mostly for volumetric rendering in general. To optimize these renders and change render quality one should know that micro volumes height (dPds) and width (dPdt) are controled by the shading quality in the mantra rop under the properties, dicing tabs and in the geometry object node under the render, dicing tab. Depth (dPdz) is controlled by volume step size in the mantra rop under properties and sampling alongside the separate control for volume shadow quality. These parameters should be set based on the scale of your scene. For instance, if your volume primitive is 100 units wide, the volumes step sizes should be increased from 1 to 10 to accomodate the size of your scene. For volume shadow quality a value of 1 will produce equal values for shadow rays and shaing rays. Volume limit on the mantra rop under the properties, shading tab sets the limit on the maximum volume bounces allowed in PBR mode.

On the geometry object containing a volume under the render tab:

- To render metaballs as volume check the parameter by this name it under the geometry tab.
- Volume filter and volume filter width adjust the filtering done on some volume primitives (Image3D, Houdini Geometry Volumes) under the shading tab. The volume filter options are box filter, Gaussian, Bartlet (triangle), Catmull-Rom, Hanning, Blackman, and Sinc (sharpening). The filter width is specified in number of voxels.

Additional parameter to add to a geometry node

- Volume density sets the uniform density of the object's volume. This value must be the same on the object as on the "Volume Cloud" shader. This is the toggle for uniform volume.
- Uniform volume sets whether to render this object as if it was a uniform-density volume. Using this property on surface geometry is more efficient than actually creating a volume object of uniform density, since the renderer does not have to march through the volume. The object's geometry must be closed (no holes). This parameter is compatible with ray-tracing and PBR, but not micropolygon renders.
- Volume velocity bounds is only necessary when using velocity motion blur with custom volume procedurals, since mantra already computes an accurate bound for built-in volume types.
- Volume isosurface ??? toggle
- Volume samples is by default one, but on a custom or complex volumetric shader you can increase this for higher quality.

Example Files

Volume Rendering - Metaballs as Volume

$HFS/houdini/help/examples/nodes/out/ifd/meta_volume_smoke.otl
Volume Rendering - From Primitives

$HFS/houdini/help/examples/nodes/out/ifd/torus_volume_smoke.otl
Reference Documents: mantra rendering properties, understaning mantra rendering, volumetric rendering, mantra rop.

Lighting Volumetrics

It is important to note you should try and just use point based spot lights with deep shadow maps when rendering volumetrics. These calculations are far quicker to compute than using any form of area light, such as, the standard line and grid light types, custom geometry light type, or environment lights. IBL with Volumetrics is very time consuming do to the increase of the amount of calculations that need to be done. Additionaly raytracing should not be done with volumetrics do to the large increase in time of rendering. These options are available and can create some desirable results, but as in most cases speed in production is more important, since they require to cast a lot more rays. Also unchecking transparent shadows which turns of deep shadow maps when depth shadow maps is on will limit the cache of data from 3-D to 2-D to be rendered for shadows leaving you with ugly non-realistic shadows.

ENVIRONMENT LIGHT OBJECT

The environment light is the tool for image based lighting, IBL, and creating realistic skies. It doesn't matter whether you need to do outdoor or indoor lighting the environment light is a good choice. It is represented on the shelf by the environment light, the sky light, and the portal light. It is preferable to use this node over the light node with the legacy environment light type. At it's base the environment light, under rendering mode direct lighting the default, is really just an inverted spherical area light that automatically scales to your scene size. This node however gives you more control than just using the light node. One of the simple controls is the clip to positive Y hemisphere. This will disregard the bottom portion of the environment sphere and only emit light from the positive Y hemisphere. For further controls that will cover you can change rendering mode, you can specify of portal for the IBL to emit through, you can specify a real world sky location, and even do cartoon like artistic ramp control.

Transform tab

The transform are limited to rotational information. You can either punch in the rotation of your map by hand or you can set the look at option to a null or other object to have better control over the positive Y axis(pole vector).

Environment Maps & isixpack

It is important to note that houdini uses it's own cubic map formatt that has been optimized for rendering with mantra. The default image formatt is .rat you can use .exr or .tiff also if your pipeline is better suited for that, but this format is limited to 1024×512 pixels and has poor anti-aliasing compared to .rat.

There are three default images that ship with houdini that you can use or view in mplay to see how they are composed, be sure to check the different channels; Right, Left, Top, Bottom, Front, and Back.

$HFS/houdini/pic/DOSCH_CARB_04SN_lowres.rat

$HFS/houdini/pic/DOSCH_DH207SX_lowres.rat

$HFS/houdini/pic/DOSCH_SKIESV2_01SN_lowres.rat

You have to use a program called isixpack that ships with houdini to convert images to this format. It is a terminal run program, with no gui, that is located in:

$HFS/houdini/bin/isixpack

Isixpack is a very picky program it will only convert vertical cross, VC, .hdr, high dynamic range, images in a 3:4 ratio to the environment map file type, Houdini Image Based Lighting, HIBL. Since most IBL librarys are latitude longitude, or you may have created your own images with a mirroball or 180 degree fish eyed lens with a nodal tripod, you will often have to convert mutliple times. Two of the better programs to work with this are HDR Shop™ and Photo Shop™. HDR Shop™ is a freeware software that is a compact program that is windows based, but can run on most emulators that will alow you to manipulate and convert HDR Images very easily. Photo Shop™ allows you to work convienently in a 32-bit workflow that allows you to touch up, blur, and manipulate your images.

Since, isixpack only accepts vertical cross images that are in a 3:4 ratio. The vertical cross image has to be layed out liked this.

top

left front right

bottom

back

You need to be careful when typing in conversion info into HDR Shop™ or another program to convert into an vertical cross image that works with isixpack. A 512 x 512 image (what the dosch series are in) will be created based on a 1536 x 2048 vertical cross image. If you type in 2048 as the horizontal width into HDR Shop™ you will get 2730 as the vertical and isixpack will throw you an error to correct the image to 2046 by 2728 to keep the final images to 682 x 682. The 3:4 ratio that HDR shop derived is technically correct, but you can not divide the horizontal evenly by three for isixpack. You do not necessarily need a very large environment map, 512 x 512 is not too small a size, just be aware of scene scale and the number of rays you may have to cast, when increasing the image size. A quick common table is;

256 x 256 HIBL needs a 768 x 1024 VC

512 x 512 HIBL needs a 1536 x 2048 VC

768 x 768 HIBL needs a 2304 x 3072 VC

1024 x 1024 HIBL needs a 3072 x 4096 VC

To convert the image open a terminal window from your operating system, the base concept is shown below and listed after that are a few examples of what was actually typed into the terminal to make it work. Just change the directory to match your own system. You can also set up an environmental variable for this program, but by default there isn't one.

Base Concept

isixpack VerticalCross.hdr HoudiniIBL.rat

Windows 7

"C:\Program Files\Side Effects Software\Houdini 11.0.504\bin\isixpack.exe" "C:\Users\personaldirectory\Desktop\MyImage_VC.hdr" "C:\Users\personaldirectory\Desktop\MyImage_HIBL.rat"

Apple Snow Leopard

/Library/Framework/Houdini.framework/Versions/11.0.504/Resources/bin/isixpack /Users/personaldirectory/Desktop/MyImage_VC.hdr /Users/personaldirectory/Desktop/MyImage_HIBL.rat

Linux Ubuntu

/opt/hfs11.0.504/bin/isixpack /mnt/server/staff/personaldirectory/desktop/MyImage_VC.hdr /mnt/server/staff/personaldirectory/desktop/MyImage_HIBL.rat

An alternative to a vertical cross .hdr is to use six images of the different cubic planes to convert to a HIBL. You must list the images in this order to derive an appropriate image front, right, back, left, top, and bottom, and they must be in the .pic format. Listed below is the base concept for this. The command can get very long when specifying the directory to all the images.

isixpack frnt.pic rite.pic back.pic left.pic top.pic bot.pic map.rat

To generate a cubic environment .rat file from a vertical cross HDR image:

isixpack kitchen_cross.hdr kitchen.rat

To modify the default size 1024 x 512 you can insert the flags -u horizontal output resolution, -v vertical output resolution, and -a turns on anti-alias the output with 3 x 3 Super sampling.

A work flow if your are using Paul Debevec's HDR Shop is to output a vertical cross and use the isixpack tool in a terminal window.

Ptex

isixpack can generate .ptx files for reflection/environment maps. These maps provide better filtering when there's lots of blurring when compared to using .rat files.

Sets the environment map (on an infinite sphere) and returns its color.

IBL Lighting Tips and Noise Reduction

When using IBL in Houdini since the environment light requires a lot of rays to evenly shade an object there a few options to do.

5. If you can use a portal geometry or uncheck clip to positive Y hemisphere this will limit the sample area for you lights casting more rays from where you want more on that later.
6. If you are turning up the diffuse limit to create secondary bounces use the indirect light to cache your secondary bounces.
7. Use the environment light only for diffuse lighting when possible and use a blurry environment map to light with. The higher the details of the image you are lighting the more the chance the rays alongside of each other are not going to be sampled from the same color causing a noise pattern. When the image is blured there is a higher chance for common color to be sampled and less apparent noise. Low sampling will allways cause noise in general, so if you make it so you notice less of it you can cheat sending out more rays.

8. Use the PBR render engines for the environment/area light since you have more control, but you can use the micropolygon or raytrace engine, but you will need to increase pixel samples alot to adjust the noise induced form the light. If you need to have that parameter up already then this may be a better option.
9. When using the PBR render engines to adjust the noise from the environment light just view your diffuse export of this one light and increase the sample quality on the light until it smooths out. You do not need to make it perfectly smooth, because a few other sample options will smooth it out to.
10. Increasing the dicing will smooth out your light noise, but it has a significant render time increase.
11. Pixel samples and ray variance anti-aliasing will have an effect, but not a major one so you should adjust your light samples first when using the light primarily for diffuse calculations.
12. To compensate for not having a reflection or specular effect when just using diffuse lighting form the environment light you can use a geometry object with a texture and phanton object checked instead. This allows you to gain control of using reflection linking in the light linkerand to gain control of your reflections. On method to do this is you can use a sphere object with a constant shader of your latitude longitude map, inside your sphere object set your primitive type to sphere, append a uvtexture sop set to polar(for spherical uving), a convert sop with divisions per a span on and set U and V to zero(cheat to normalize your uvs and convert it to poly so the render engine does not have to sub-divde you geometry), and a reverse node(to flip your normals in). Then uniform scale and postion your sphere object to surround your scene/object. A long process, but it gives you control especially with high detail images, also you can swap out your reflection maps until you get the prettiest. You can alternative use a box to for this process, you just need to spend more time adjusting your uvs.
13. Make sure if you do the prior method you do not use a nurbs sphere, as expalined in the geometry light type section, geometry that is sub-divided creates more noise. So it is best to use a low polygon object.
14. To adjust the noise created from the reflection map in the PBR rendering engines increase the max ray variance anti-aliasing vlaue, especially if you have increased your reflection or refraction angle to blur the image. This will also smooth out the light noise, but not as quickly as sampling quality on the light.

Latitude Longitude

If you want to use a latitude longitude map in houdini to do IBL you have to use a geometry light with an inverted sphere. Use the method described prior in part 8 of the IBL Lighting tips to create your geometry then use your light. Any section of the object that you can use a carve sop or cliping sop to remove will help, also by cacheing out the object, writing it to a .bgeo and just using a file will lead to quicker load times. Look at the geometry light for more info to set up this light type. This method is not as optimized as using houdini's environment map, so if you want better results at the cause of more time you can do this. Although perfecting this for your pipeline and making a digital assets of it will speed up your work flow.

Render Options tab

Rendering mode gives you the option of direct lighting, ambient occlusion, and ray tracing background. Direct lighting is treated as inverted spherical area light that encloses the scene. When used mantra will send more samples toward brighter portions of the environment. It will also enable variance antialiasing when rendering with PBR or the Surface Model VOP. To adjust the sampling threshold adjust the min/max ray samples to automatically detect and reduce noise in the lighting. Direct lighting is good for IBL with directional light sources, when you need specular contribution from the IBL, and when you require per-light exports. The sampling quality parameter is enable for this option and works like a regular area light. Also the portal geometry is enabled, but will cover that in a momment.

Ambient occlusion, AO, is computed using the VEX Global Illumination shader, and will send evenly distrubted rays into the hemisphere to check for blocking surfaces it takes the avergae of these and darken the pixel. This will only contribute to diffuse lighting if used in a scene with other lights or as a AO pass on its own. It is best to not use an environment map with this or if you do, one that is very uniform and not intense. This light is good for irradiance cacheing not using PBR, since PBR does not function with irradiance cacheing in the mantra rop. The sampling angle and adaptive sampling parameters will be enabled with this option, allowing you to set the half-angle to send rays for irradiance/occlusion, and an automatic sampling optimization that reduces the number of rays cast, respectfully. The latter reduces the number of samples when there is little variation in occlusion above the sample point. This can improve performace when your samples go above 64 samples per a pixel, but may induce some possible flickering or additional noise.

Ray tracing background is for indirectly contributing to the scene when loose rays exit the scene. This is similar to the environment map in the reflection area of most shaders. This mode is supported by the resolvemissedray VEX function in shaders like the Surface Model VOP, and PBR rendering. This render mode is good when rendering with out an environment map or if you do, one that is very uniform and not intense like the AO render mode. Additionaly, when using PBR, a shader that uses a lot of raytracing for indirect lighting, or when using the indirect light to calculate irradiance, speeding up lighting for secondary rays.

Portal Lights

A portal light is an environment light with a piece of geometry selected in the portal geometry parameter. What portal geometry actually does is specify an area for the renderer algorithm to focus the sampling on, in an enclosed environment, optimizing the distributed rays. The portal geometry should be a single sided piece of geometry that is not renderable so that you do not multiply the intensity or create shadow artifacts, like a geometry light, since the light is based on surface normals. A correctly configured portal should produce the same rendered result as when the protal is diabled, but with much less noise. Portals should normally not be used for outdoor scenes where the environment map is fully visible, or when the environment map contains sharp variations in intensity, since the the standard direct lighting algorithm may produce better results. If you are using a blured map for diffuse lighting only you can roughly trace out the area where the light should be sampled from, use some color manipulation in cops and a trace sop to help define these areas.

Two examples, would be the pantheon and a warehouse. The pantheon is a domed building in Rome that has a central opening(oculus) to the sky, and a warehouse, that has windows, is often a large structure with many and large windows to the sky. If you were doing the Pantheon with the portal light you could use a nurbs sphere scaled correctly, a carve node with only the second V checked on and set with a low number, with a reverse sop that will flip the normal. For a warehouse, you would place a two by two grid to fill each window and sky light, not necessarily each pane, and set the normals to face in to the set.

Sky Environment Map tabs

The ability to do this node has been in Houdini for a while. It is a combination on the separated environment light and the compositing node known as the sky environment map. You can actaully dive inside the light and see how this is made. With H11 the color information provided here became an excellent source to produce light information for their improved area light especially the inverted spherical environment light.

Under the Sun tab you have multiple ways to orient the vector for the sun to be in the sky. The use parameter lets you do Rotate Angles, Direction Vector, Azimuth and Elevation, Location Date and Time, and Location and Fractional Day of Year. Here are a few tidbits on the parameters that stray from the obvious. Rotate has the -Z axis aligned toward the sun. Direction Vector is the explicit vector that points towards the sun. Azmiuth the default plane (C) shows the right view, so 90 will show the sun centered in that view. Latitude and Longitude are from the postion of the observer. Day of the Year, .5 represents noon which is convenient for animating and expression functions.

The Sky tab gives you the use option of between realistc and ramp. When realistic is on haziness and brightness control how sunlight scatters in the air and the light intensity of the sky. Ramp is a good option to use in a more artistically designed endeavour. Also when ramp is selected you can manually adjust the ground tab parameters to adjust the normal, rotation, and elevation of the ground bounce color. If you want the effect of the ground make sure clip to positive Y hemisphere is unchecked.

Reference Documents: Environment Light object, Lighting Environment/reflection maps, reference isixpack, Sky Environment Map compositing node

INDIRECT LIGHT OBJECT

The indirect light is good for caustics and diffuse lighting a.k.a. final gather, FG, indirect global illumination, GI, or color bleed as described in other render packages. The light is designed for use with all render engines. In prior versions of Houdini photons were created by a second mantra rop. This is no longer supported and the new system is a much better workflow. The indirect light will override the settings in PBR in the mantra node, and for non-PBR it will add diffuse lighting that is not set by the render engine.

Indirect Light Render Tips:

15. You can create a indirect light by eith LMB the shelf GI Light, or by using the tab menu to create an indirect light.
16. To verify that photons are generated during the initial render you can view it through the Render View and the photon generation progess will be reported at the top right. Make sure to press render to see the progress each time, or else it reads from the cache made previously when it refreshes. One can also turn on the statistics in a mantra node to verbose level 1 when rendering to mplay to read it in a terminal.
17. To verify that photons have been generated after your first render you can view the .pmap or .pc with a file sop and Houdini will convert them to points you can see in the viewport with display points on from the right stow bar.
18. To verify that photons have been generated visually switch the light type to direct global photon map to see them rendered directly. One can also when rendering in PBR on a mantra rop create an image plane of direct diffuse and check export variable for each light, under the properties, output tab.
19. A good way to test if enough photons exist is if you switch between indirect global photon map and direct global photon map the overall light level for each render should be the same.
20. If the render has large blocks of constant illumination, increase the photon count until the quality reaches an acceptable level.
21. If you notice some artifacts in geometric corners, increase the prefilter samples or try increasing the photon distance threshold. Increasing the number of samples may require you to go back and increase the photon count to sharpen the photon map result.

Indirect Light Light Types

There are four options under light type; indirect global photon map, direct global photon map, caustic photon map, and irradiance only.

- Indirect global photon allows for secondary indirect lighting when the diffuse limit on the mantra rop is set to at least 1. This will produce a photon map that with PBR on will speed up indirect lighting more than when just using the diffuse limit. What happens is that rather than using the photon map directly, diffuse rays will be traced from the shading sample and the photon map will be used for these secondary hits. The photon map will decrease render time and reduce noise, to test this check the light enabled toggle.
- Direct global photon map is for visual testing of the photon map. This is generaly not used for renders.
- Caustic photon map allows you to do caustics, or patterns of focused reflected and refracted light. This option will configure the photon map generation algorithm to only calculate light paths that reflect or refract from objects before being absorbed by a diffuse surface. For more info read the rendering caustics section.
- Irradiance only produces indirect diffuse lighting using path tracing. This is only useful for non-PBR render engines, mircopolygon and raytracing. It will use the irradiance VEX function, therefore you can cache the info in the irradiance tab of the mantra rop.

Photon Map Options

These are your main control setting for indirect lighting and caustics no matter the render engine. The first option is the last one you need to adjust. That is once you have your caustics positioned and optimized you should uncheck auto-generate photon map and save render time so you don't have to recalculate your cache each render. Following that parameter the photon count is the approximate number of photons that are cast into your scene. If you use the file sop method to view your photons you can actually get a count of the photons that have been cast if you do not prefilter your photon map by MMB on the file node.

Light mask should usually be specified, by default a wildcard, *, will select all lights in your scene to cast photons from. In a large scene this can slow your renders down siginificantly. So it is often best to specify a light. You can uncheck light enabled on the specified photon casting light if you want, but naturally a key strong enough to cast caustics will have strong light and shadows so unless you have a strong artistic reason to cheat this you may want to leave it on. Additionally for each light you can setup an indirect light for it and with an image plane set for direct diffuse and export

variable for each light so you can have composite control over each light.

Photon target is a specified object that you want photons to interact with to optimize your render. The photons will not cache on this object they will just bounce off or pass through it. For instance, if you are rendering a simple scene with a glass of water in it, you specify just the glass and not the environment around it. If you did not specify an object, or specified too many objects, and your render does not show any photons and produces an error indicating photon map generation was aborted, refine your caustic receiving object and/or lights emiting, or break them up into separate lights, objects, and indirect light, with light linking.

Photon file is the directory where the file is generated to disk. When rendering an animation it is good to specify the frame number, $F, or when doing per light caustics you can specify the node name, $OS or`opname(".")` in the file directory or file name.

Photon map stores direct lighting, normally you should leave this option on and it will store the direct as well as the indirect lighting. For instance, when caustic photon maps is on this setting will be ignored so that only the indirect is stored.

Filtering

Filtering for photon maps can either be a pre or post process, prefilter or filter samples, respectively. You can pre process the photons so that while the photons are generated you cull the data, so you do not have as many photons stored on disc, good for a large project. If you post process the photons all of the photons are written to disc and then you just filter them when you load the photons up to be read, good for small project or flexibility. To save iteration time when you are trying to reduce noise once you have the photons all cast in the position you want you can slowly turn down or up the filter samples to adjust the render.

The photon filter has two settings convex hull and sphere. Convex hull improves photon map quality in geometric corners. Sphere will decrease your render time at the expense of accuracy. So get your indirect light to where you like it, then switch to sphere and see if you like the results Filter samples is the post process filtering. When rendering with a prefiltered photon map, you should use a small number (generally from 1 to 10). When rendering non-prefiltered photon maps, you should use a large number (generally from 50 to 500). Prefilter samples if you use this option the range should normally be a large value (generally 50 to 500). Prefilter ratio is the ratio of prefiltered photons over photons that were initially sent out before cacheing. Ratios of less than 1 can improve performance while introducing only a small difference in downward quality.

Indirect Diffuse Options

These sampling options only apply when using a non-PBR render engine. When rendering with PBR you should use the sampling controls in the mantra rop such as pixel samples and min/max ray samples. Further these settings are only enabled for the light type indirect global photon map and irradiance only. Sampling quality works much like it's area light counterpart by sending out more rays for sampling to smooth noise issues. Max ray distance when enabled sets the distance from the sample point to consider geometry for irradiance. Sampling angle is the half angle over which to send rays for irradiance. Adaptive sampling can be used when the sampling is higher than 64 samples. It is an automatic optimization that will reduce the number of samples when there is little variation in occlusion above the sample point. This will improve render time, but may induce some flickering and additional noise.

Photon Distance Threshold

The photon distance threshold is a quality/performance control. When too few photons are present in a photon map, the distance between photons can be significantly larger than the distance that an indirect lighting ray travels, which can lead to artifacts in the render. This parameter indicates that when the indirect lighting ray travels a shorter distance than the distance between photons in the map, that the illumination should be computed directly (via path tracing) rather than using the photon map. Larger values will favor more path tracing while smaller values will favor the use of the photon map. A value of 0 indicates that the photon map should always be used, potentially leading to artifacts in corners, but decrease render time. Generally a value between 0 and 4 is reasonable, with a value of 4 eliminating the vast majority of artifacts in geometric corners, but increases render time.

If your photon map contains very smooth illumination (for example, when using many filter samples), you can improve performance by decreasing this parameter to 0. If you photon map is poorly filtered, you can improve quality by increasing this parameter to a value larger than 1.

To visualize how much the photon map is contributing to the image relative to path tracing:

22. Configure your rendering engine to Physically Based Rendering.
23. Add an indirect_diffuse deep raster and enable Export variable for each light.
24. Look at the deep raster plane for the indirect light and for your other lights. You should see that in corners of your geometry, the indirect_diffuse will have a larger contribution than other lights. As you decrease the photon distance threshold to 0, the contents of indirect_diffuse for lights other than the indirect light should drop to 0.

Rendering Caustics

Caustics can seemingly be a tricky thing to generate even after knowing how the indirect light functions. Caustics are defined in houdini by the material of the object and the position of the light. So the most basic setting to generate caustics in houdini is to create an indirect light and set the light type to caustic photon map and make sure the object you want to have caustics has reflections or refractions in the material. If you have done this and you don't see caustic patterns in your scene there a number of reasons it may not have worked so the best thign to do is to break down the scene to it's most simplest components that are causing the caustics whether you're starting from scratch or you are an experienced user. A good example to test this with and see how the shot is compiled is to

The best thing to do if you are starting from scratch or an experienced user is to go to the help documentation and load the example file, tubecaustics.otl. This scene shows a very good simple example of how to generate caustics that can be disected to problem solve your own potential problems.

To build the scene yourself here is a list of steps;

25. Create a shopnet(shopnet1)
26. Create a **mantra surface** in shopnet1 - name ground, uncheck Reflect Lights
27. Create a mantra surface in shopnet1 - name metal, uncheck Diffuse, check Reflect Objects,
28. Create a camera - translate 12,13,12, rotate -39,45,0,
29. Create a ropnet(ropnet1)
30. Create a mantra rop node - name pbr, Extra Image Planes 1, Vex Variable direct_diffuse, check Export variable for each light, Rendering Engine Physically Based Rendering
31. Create grid (Ground) - apply to ground
32. Create tube (Object you want to cast Caustics) - apply to metal
33. Create light - translate 3.5,2,-9, rotate -45,135,0, Light Intensity .2
34. Create indirect light - Light Type Caustic Photon Map, Photon Count 500000, Light mask hlight1, Photon Target Tube, Filter Samples 100, uncheck Prefilter Photon Map
35. Create environment light - Environment Map $HFS/houdini/pic/DOSCH_SKIESV2_01SN_lowres.rat
36. Create indirect light - Light Type Caustic Photon Map, Photon Count 500000, Light mask envlight1, Photon Target Tube, Filter Samples 100, uncheck Prefilter Photon Map

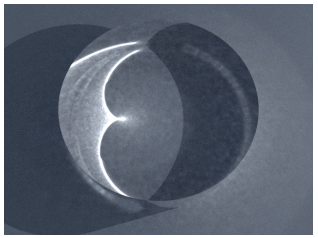
Make a mantra render to mplay to see the scene rendered. Take a look at the different image planes that were output in the channels menu as seen on the left. These are exports of the different lights, the point light, the environment light, and the two respectively connected indirect lights. You can see what is actually being added to the scene from the photons of the indirect lights, as opposed to just the regular light emissions. This a good test to see if the photons have been emitted at all in your own scene, and two if they are actuaslly hitting your object from your lights source.

If you move the light the caustic pattern will change, and if the light is not in the right position you won't have caustics at all, which can often be the case. So when you set up your scene use a low photon count for quick renders and test and adjust the position of your light to

Photon maps will only be generated for point lights and spot lights. If you are using point lights, make sure it is close enough to the scene to ensure that enough photons will hit your transparent geometry. If you are using spot lights, make sure it is illuminating as small an are of your scene as possible. You can do this by adjusting the Cone Angle and Cone Delta parameters on the Spot Light Options sub-tab of the Light tab.

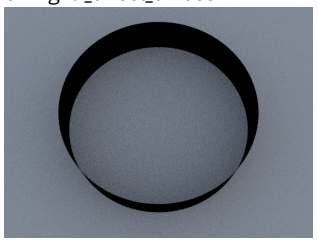
TUBE CAUSTIC OUTPUT:

Tube Caustics Color



pointlight1_direct_diffuse

envlight1_direct_diffuse



envlight_photons_direct_diffuse

pointlight_photons_direct_diffuse

If your render produces the error message: “Aborting photon generation from light...”, try moving your light closer to the scene or fixing the orienting toward your geometry. If the light is too far away from your scene it will not be able to efficiently generate photon maps.

When setting up caustics it is good to set the light mask to the lights you want to cast photons so your photons are concentrated in the right places.

If you render a scene that requires both caustics and indirect diffuse lighting, you should render with at least two indirect lights, one specified for indirect diffuse the other for caustics.

Reference Documents: Indirect Light object, Setting up physically based rendering, Mantra Rop

Example Files : TubeCaustics, IndirectLightBox

MANTRA ROP

Overview

A mantra rop is a render output driver in Houdini. Mantra is houdini's native set of render engines, additional there are output drivers for 3Delight, Pixar's Renderman, RenderDotC, Sitex's Air, and Mental Ray. First we are going to cover the different mantra render engines and break them down via their geometric sampling and shader sampling properties, followed by the parameters that are specific to them, and cover the rest of the parameters.

Basic ROP Workflow Settup

This is al isted process of how to quickly set your mantra rop before you start optimizing.

37. Set the frame range, Valid Frame Range to Render Frame Range, adjust Start/End/Inc
38. Set the camera path, Camera, under main tab
39. Set the image output directory and format, Output Picture, under properties and output tab
40. Set the render engine, Rendering Engine, under properties and render tab

Mantra Render Engines

Under the properties, render tab in the mantra rop is the menu for rendering engine. Listed there is micropolygon rendering, ray tracing, micropolygon phyiscal based rendering, physical based rendering, and photon map generation. Photon map generation has been outdated by the indirect light so we'll consider that legacy, leaving four render engines.

Render engines in mantra can be broken into four steps; the geometry refinement process, the shading sampling process, the sample filter process, and the pixel filter process. The render engines are best defined by the first two of the four steps they take to create a pixel as all four engines share the last two processes. There are only two geometry refinement processes in Houdini, micropolygon and raytracing. While the shader sampling process has three processes micropolygon, raytracing, and physical based rendering (PBR). PBR is only a shading process which is why it is paired up with micropolygon and ray tracing. So technically and as defined by variables the render engine physical based rendering is really raytrace PBR.

The simple explanation of the four step rendering proces in mantra is that during the first step scene geometry is refined into small chunks of primitives. The second step the shader is run a certain amount of times across a pixel to create a series of direct and interpolated samples of the color. The third step these samples are composited into a single color for a pixel. The fourth step these pixel colors are filtered across several pixels to determine the final color.

In order to best break down the mantra rendering process to understand how the mantra render engines work. We are going to break down the four steps first into details and then tell you what parameters you can adjust specifically for that engine. We will then cover some general render optimization processes. Followed by breaking down the common parameters of the mantra render node in order of appearance. For the first two render steps we will breakdown the geometry refinement process and shading sampling process based on the different render engines. Parameters or functions that are repetative we will just cite back to previous explanations, so instead of explaining the micropolygon refinement process twice for non-PBR and PBR rendering we will only explain in once. So raytrace PBR will seem very short on info.

Micropolygon Rendering

Geometry Refinement

Micropolygon rendering is based on the REYES algorithm. In which it refines large geometry primitives, for instance, a single four sided square that is nurbs or sub-ds, by splitting them into smaller and smaller primitves the size of a pixel, it will further "dice" larger primitives into a grid of micropolygons to make it the size of a pixel. Polygons smaller than a pixel do not get affected, so it is best

to use sub-d, or nurbs surfaces for these shapes instead of having 10 million polygons in the space of a pixel. Subsequently, REYES works best with quad meshes for this workflow.

Whereas micropolygon rendering refines and dices surfaces into micropolygons (each with a size measured by dPds and dPdt),

Shading Sampling

Mantra runs the shaders once at each vertex of a micropolygon grid. It then interpolates samples randomly across the pixel as defined by where the shader was ran at each vertex of a micropolygon. At the minimum the micropolygon engine will run the shader at least once per a micropolygon depending on the size, diced quads that take up the whole pixel will get roughly the 4 samples, while for multiple polygons that fit within a pixel the shader will be run for each of these polygons so 10 million polygons within a pixel means 10 million shading process.

The shading process is run at the begining of each frame, and then the additional interpolated samples are taken over time, across the shutter opening, giving fast and less accurate motion blurthan raytracing. Shadows on non-moving micropolygon will not blur because of this. To fix this you can check on raytrace motion blur on the mantra rop under the properties, sampling tab. This will distribute secondary rays sent by the shader across time, allowing you to achieve the same look of raytraced motion blur with micropolygon rendering.

Micropolygon Parameters

Best Quality Control - Min Ray Samples

Secon Best - Dicing

Tile size - Mantra dices polygons into micropolygons in batches called buckets. Because each bucket is rendered independently of its surroundings, the renderer can throw away all the computed micropolygon geometry for the bucket once it's been shaded, saving memory.

Micropolygon measuring are the three measuring devices to determine if primitives are "too big" and need to be diced into micropolygons. Uniform measuring generates uniform divisions. The size of the divisions are controlled by the shading quality on the mantra rop, this is a global multiplier. Locally there is the shading quality parameter at the object level on geometry nodes under the render, dicing tab. Raster space measuring measures geometry in screen space. This is roughly equivalent to the "nonraster -z 0" measurer, so it has been deprecated in favor of the following approach. Non-raster measuring measures geometry in 3D. The z-importance is a bias for the z-component of the surface. A z-importance of 0 means that the x and y components of the object will be the only metric in determining the size of the object. This is roughly equivalent to raster space measurement. By increasing the z-importance to 1, the z measurement becomes more meaningful. It is possible to increase the z-importance beyond 1.

If you think of a flat primitive in the XY plane, the z-importance has no effect. However, if a primitive is nearly in the XZ plane, z-importance has more influence on the dicing. With a z-importance of 0, only the projected measurements will be used, which will result in long, thin strips being created. With a z-importance of 1, the grid will be more uniformly sub-divided. With a value greater than 1, more divisions will be performed in Z. This is important when displacement mapping is being performed.

The reflect limit and the refract limit need to be placed on the shader for this, and they are not the ones on the mantra rop which are the controls for the PBR equivalent. This sets the max number of bounces per a reflection and refraction rays. The refraction limit is normally pretty low by default, 1, so for glass and such you want to add this parameter and increase it, while the reflection limit by default is set at 10.

Dicing flatness under the render, dicing tab on the geometry object node. This property controls the tesselation levels for nearly flat primitives, by increasing the value more primitives will be considered flat and will be sub-divided less.

Ray Tracing

Geometry Refinement

Raytracing has algorithms that do efficient raytracing of points, circles, spheres, tubes, polygons, and mesh geometry. Other surface types, e.g. subdivision, NURBS, and any surface with a displacement shader are refined into micropolygons, and the renderer intersects the rays with the micropolygons. To optimize your render, if you have a scene with no nurbs, sub-d, or displacement raytracing

will be much faster than micropolygon.

Shading Sampling

Raytracing runs the shaders on every sample as defined by the pixel samples parameter. So if the number of pixel samples is 3 × 3 the shader will run 9 times. Raytracing will often (but not always) run the shaders more often than micropolygon and will be slower, but raytracing gives more accurate results, because the shaders are run independently on each sample. Raytracing will take more time too if the shader sends out a large number of secondary rays to do reflection, occlusion, etc., but you can cache these with the irradiance function on the mantra rop or in the indirect light.

The shading process is run on every sample so the shaders are run at different sub-frame times, possibly with the geometry in different positions, giving accurate anti-aliased motion blur.

Raytracing Parameters

The dicing parameters for the raytrace engine follow the same concept as the micropolygon engine, but contains it's own parameter set. These are similarly named and placed just with the ray header; ray shading quality on the rop, ray shading quality on the object, ray measuring, and ray z-importance.

Ray tracing acceleration is a spatial data structure used to optimize the performance of ray intersection tests against complex geometry. There are two types that can be used; KD-Tree and bounding volume hierarchy. Normally, the KD-Tree will produce the fastest raytracing performance at a modest initialization time than using the bounding volume hierarchy, but sometimes you should switch it up since the alternative can be faster.

KD-Tree memory factor changes the memory/performance tradeoff used when constructing KD-Tree acceleration data structures. Values larger than 1 will cause mantra to use proportionally more memory and take longer to optimize the tree in an attempt to make ray tracing faster. Smaller values will cause mantra to use proportionally less memory and take less time to optimize the tree while possibly compromising ray tracing performance. The default value of 1 will try to balance the amount of memeory used by ray tracing data structures with the amount of memory used by geometry.

If you are noticing long tree construction times (in IPR, the render view pane hangs for a long time in the "Building Octree..." Phase), try decreasing the KD memory factor 0.1. If your render is too slow after tree construction, increase the value until you find a good balance of tree construction time vs. render performance. Additionaly you can see the "Building Octree..." hang in the verbose statistics.

The reflect limit and the refract limit are the same parameters as in micropolygon rendering on the shader, but are usually applied to the shader.

Ray predice under the render, dicing tab on the geometry object node will cause this object to generate all the micro-polygons before the render begins. Ray tracing can be significantly faster at the cost of potentially huge memory requirements. When ray-tracing, it's more efficient to pre-dice all the geometry in the scene, rather than caching portions of the geometry and re-generating the geometry on the fly. This is especially true when global illumination is being computed (since there is less coherency among rays). The options are disable predicing, full predicing, and precompute bounds. Full predicing will keep everything in memory, while precompute bounds will define the area to be diced and tosses out the geometry it doesn't need. This is currently not supported for per-primitive material assignment (material SOP).

Micropolygon Physically Based Rendering

Geometry Refinement

Same as Micropolygon process

Shading Sampling (PBR Explained)

In PBR secondary rays are controlled by the surface's BSDF (Bidirectional Scattering Distribution Function), where as in traditional shading the shader controls the number and distribution of rays. A BSDF is a function that takes the incoming light angle and a reflected light angle, and returns the magnitude of the reflected light given that "bounce."

mantra runs the shaders at the vertices of the micropolygon grid. However, because PBR depends on taking many shading samples, it runs the shader many times (controlled by the Max ray samples parameter), sending rays in random directions controlled by the BSDF. As with regular micropolygon rendering, it then takes samples across the micropolygon. The values of the samples are interpolated from the values at the corners (where the shaders ran). The number of samples is controlled by the Pixel samples parameter.

Best Shading Control - Max Ray Samples and Sampling Quality

PBR shading

PBR (physically based rendering) is a shading process that supports accurate physical simulations of lighting, shadows, and shading.

PBR automatically enables: raytraced reflections and refractions, glossy reflections and refractions, and reflected light sources.

PBR is useful for rendering many types of scenes, for example: scenes that requires soft shadows from an environment map or area lights, lots of indirect lighting (such as closed rooms with reflective walls), glossy reflections and refractions, and caustics.

PBR is a shading process. It can work with either of Houdini's sampling methods (micropolygon sampling or raytracing). That's why the mantra node lets you choose Micropolygon PBR ("Micropolygon Physically Based Rendering") or Raytracing PBR ("Physically Based Rendering") using the Rendering engine parameter on the Sampling sub-tab of the Properties tab.

Whereas in traditional shading the shader programs control the number and distribution of secondary rays, in PBR secondary rays are controlled by the surface's BSDF (Bidirectional Scattering Distribution Function).

In PBR, the surface shader is responsible for computing the BSDF of the surface. Whereas traditional surface shaders set Cf (surface color) and Of (surface opacity) color values, PBR shaders set F to a bsdf value (bsdf is a new VEX datatype introduced in Houdini 9).

A BSDF is a function that takes the incoming light angle and a reflected light angle, and returns the magnitude of the reflected light given that "bounce".

For each shading sample...

Mantra computes the direct lighting contribution for all lights in the scene. This involves sending up to one shadow ray toward every light in the scene, as well as one additional ray to account for reflections of light sources.

Mantra takes the eye angle, and a random direction based on the shape of the BSDF (for example, in diffuse BSDF, every secondary ray direction is equally likely, while in a reflective BSDF, the reflection direction is much more likely) and sends an indirect lighting ray. If the indirect lighting ray hits a surface, mantra runs its shader and samples its contribution to the current surface's color.

Running the indirect lighting surface's shader will send out secondary rays for that shader, and those rays may hit surfaces, causing their shaders to run, sending out more rays. To prevent this process from continuing forever, mantra uses the reflect limit settings (on the PBR subtab of the Properties tab) to limit the number of bounces mantra will follow.

So, for each shading sample in BSDF, mantra will only generate a few additional rays: direct lighting rays in the direction of light sources, a direct lighting ray for reflections of light sources, and an indirect lighting ray in a direction chosen via the BSDF.

The output of the BSDF for the incoming eye direction and the direct and indirect lighting rays is the final surface color.

Because each sample only traces a few rays, different samples might compute completely different colors. For this reason, in PBR you will typically use 16×16, 32×32, or more samples per-pixel (pixel samples setting). Accumulating that many samples will average out the differences from sampling only a few rays.

The rendering engine will only execute the code needed to compute the appropriate variable(s), so in the traditional shading pipeline, the renderer will run lines 3 and 4 of the above code, while in the PBR pipeline it will only run line 5. This also means that if your shaders aren't set up to compute F and you switch to PBR rendering, the shaders will output black, because the traditional shading code isn't run.

Micropolygon Physical Based Rendering Parameters

Pixel Samples

Min Ray Samples

Max Ray Samples

Raytrace Physical Based Rendering

Geometry Refinement

Same as Raytracing process

Shading Sampling

Pixel Samples

Min Ray Samples

Max Ray Samples

Raytrace Physical Based Rendering Parameters

Sample Filtering

image:deepresolver = (‘’)

When generating an image, mantra runs the sample filter to composite samples to a single color. Mantra then runs the pixel filter to produce the final color for a pixel. A deep resolver is used to store information about each sample prior to sample filtering. This allows the image resolver to store information about each individual sample before compositing. The image:deepresolver property specifies the resolver and any arguments to the resolver.

Options:

- filename (default = “”) – The filename to output the deep shadow information.
- ofstorage (default = “real16”) – The storage format for Of. The value should be one of...
- real16 – 16 bit floating point values.
- real32 – 32 bit floating point values.
- real64 – 64 bit floating point values.
- pzstorage (default = “real32”) – The storage format for Pz. The value should be one of...
- real16 – 16 bit floating point values.
- real32 – 32 bit floating point values.
- real64 – 64 bit floating point values.
- ofsize (default = 3) – The number of components to store for opacity. This should be either 1 for monochrome (stored as the average value) or 3 for full RGB color.
- compression (default = 4) – Compression value between 0 and 10. Used to limit the number of samples which are stored in a lossy compression mode.
- zbias (default = 0.001) – Used in compression to “merge” samples which are closer than some threshold.
- depth_mode (default = “nearest”) – Used in compression to determine whether to keep the nearest, the farthest or the midpoint of samples. The possible choices for depth_mode are...
- nearest – Choose the smallest Pz value.
- farthest – Choose the largest Pz value.
- midpoint – Choose the midpoint of Pz values.
- depth_interp (default = “discrete”)
- discrete – Each depth sample represents a discrete surface.
- continuous – Each depth sample is part of a continuum (i.e. volume).

Example: shadow filename test.rat ofsize 1

Pixel Filtering

Filtering happens after the shader process is run for each sampleand prior to the quantization of each pixel, it sums the contributions of each sub-pixel sample to get the final pixel color. It then quantizes the floating color vector into the specified color format (e.g. 8-bit or 16-bit integer) to get the final value of the pixel in the C (color) plane of the final image. The pixel filter on the output tab is the filter control. It is a string input which is defined by the filter and the pixel height and width to sample from. Increasing the filter width will increase anti-aliasing/blurring. The drop down menu includes several preset; Unit Box Filter(box 1 1), Gaussian 2x2(gaussian 2 2), Gaussian 3x3 "softer"(gaussian 3 3), Bartlett "triangle" "cone"(bartlett 2 2), Catmull-Rom(catrom 2 2), Hanning(hanning 2 2), Blackman(blackman 2 2), Sinc "sharpening"(sinc 2 2), Closest Sample Filtering(minmax min), Farthest Sample Filtering(minmax max), Disable Edge Antialiaisng(minmax edge), Object With Most Pixel Coverage(minmax ocover), Object With Most Pixel Coverage "no filtering"(minmax idcover)

Alternatively instead of running filtering you can check Sub-Pixel output and each sub-pixel is output as an unfiltered pixel. The final output resolution will be scaled by pixel samples to determine a new final output image resolution. This is good for composite testing, especially when matching between different render engines.

For instance, if you rendered at 1080p (1920 x 1080) and your pixel sample size was 8 x 8 your final resolution would be 15360 x 8640 image:resolution was (1024,512) and image:samples was (4,6), the image rendered would have a resolution of 4096 by 3072. Each pixel would represent a single unfiltered sub-pixel sample.

Standard Render Optimization

Inheritance

Inheritance is a level hierarchy of parameters in which lower level parameters take precedence over higher level parameters. The render hierarchy goes renderer defaults, output driver, camera, object, and primitive. For instance in H10, dicing was normal adjust per an object, but if you wanted to you could remove the dicing commands from the geo and place them on the output driver to control all object level dicing.

Geometry Sampling

Refined geometry is cached during raytracing, and there is a limit to the amount of refined geometry mantra stores for raytracing. The cache size is controlled by the geometry cache size parameter.

This is important to remember because refinement requires more memory. If you can reduce the amount of geometry in the scene that requires refinement, the raytracer will use less memory. Refinement requires the creation of duplicate, independent refined geometry. When raytracing geometry that requires refinement (such as subdivision surfaces or polygons with a displacement shader), try using simple base primitives, such as lower resolution meshes, to reduce memory use. Refining instanced geometry may create independent refined geometry for each instance, defeating the memory savings of instancing. (Subdivision surfaces and NURBS may still share some data if the instances require the same detail level. Only displacement requires wholesale duplication of each instance.)

Dither

Dither determines the noise added to an image prior to creating the final output pixels. Dithering is the addition of low-level noise to colors before they are converted from high precision floating point numbers into 8 bit (0-255) integers. You can avoid Mach Banding in an image containing slight luminance changes by adding approximately 0.004 (1/255) noise to them.

Normally, dithering is done in HSV (hue, saturation, value) color space in order to provide better dithering for highly saturated objects.

Pixel Sample Sub properties

Shading Sampling

If the vm_smoothcolor property is off, the sample simply gets the value from the bottom left vertex instead of interpolating. This might be useful if you're sampling plate geometry and don't want interpolation across its face.

Jitter

A floating point value which adds noise to pixel sampling patterns. A value of 1 will fully randomize pixel sampling patterns, while a value of 0 turns of pixel jitter resulting in stairstep artifacts when too few pixel samples are used. Jitter only applies to pixel antialiasing and does not apply to motion blur or depth of field sampling (which are always randomized).

Random Seed

An Integer value that controls initialization of pixel sampling patterns. Different random seeds will prodcue different pixel sampling patterns.

Because the Pixel samples parameter controls how many sub-pixel samples are averaged together to get the pixel color (whether the shaders run at the samples or the values of the samples are interpolated from the vertices), increasing the pixel samples gives better anti-aliasing for geometry edges, motion blur, and depth of field.

Samples are distributed randomly in space (across the face of the pixel) and in time (across the shutter open time of the frame, during which time the geometry may move).

Noise Level for Ray Variance Anti Aliasing

Noise threshold used for ray variance anti alaising. Lower values produce less noisy images

The variance threshold to send out additional anti-alising rays. When nearby samples are very similar, fewer anti-aliasing rays will be sent out. When nearby samples are different, more rays will be sent.

PBR

Basic Render Parameters

Header

Main

Objects

Output

Motion Blur

DOF

Irradiance

Reference Documents: Mantra render node, Rendering Expressions in file names, Setting up physically based rendering, Render quality and improving render time, Advanced dicing control, Rendering extra image planes(deep rasters), Rendering understanding mantra rendering, Rendering control dependencies between renders, Rendering with a large number of polygons, SOHO, Rendering an image, Rendering FAQ, Properties Mantra 9.1 and 9.5 rendering properties, Mantra rendering properties, Shader (SHOP) properties, Rendering properties, Rendering subdivision surfaces

MANTRA ARCHIVE

WIREFRAME RENDERING

Wren ROP

The wren output driver is designed to produce rendered wireframes from houdini. This is a much better method than producing a wireframe via a flip book since you will have nicely aliased renders and a lot more control.

By default the renderer will only render silhouettes of the image. In order to produce wireframes of all the edges append a facet or fuse sop to your geometry and check unique points. Since it renders edges that have unshared points. The facet node in general will allow you to create some unique renders out of wren. They will create an '80's hidden line vector look good for model turn arounds and breakdowns.

If you want to see the wires on the back side of your objects append an end sop and under the close U and close V parameter pick unroll. This process will open up the primitives and remove the poly faces.

Mantra ROP

To render a wireframe from the mantra rop the mantra rop takes in the linewidth attribute **width**, if present, to determine the width of the line. So after using an end sop to unroll the poly faces you can apply an attribute create sop under the name parameter label it **width** and in the value parameter modify the first value to your rendered line thickness, try .01. Then you can apply a color sop to change the color of the line. This method will allways render the back side of the poly.

Additionally you can set your wireframe width and base it on depth to adjust the near and far size. Apply an end sop to unroll the poly faces to your base geometry. Followed by a uv texture sop, set the texture type to perspective from camera, create and or set your camera and define it in the camera parameter. Append an attribute create sop and under the name parameter label it **width** and in the value parameter type in **8e-5*max(1e-9,abs($MAPW))**.

Statistics

The first thing to know about statistics is that they are shown in a terminal. In windows you get this for free when the Houdini Console pops up when you have statistic setting turned on. For Linux you need to launch houdini from a terminal to have this info displayed, and it is in that termial that you launched Houdini from you can see the info.

There are three information parameters under the statistics tab, Verbose Level, VEX Profiling, and Alfred Style Progress.

Alfred Style Progress when checked shows the percentage of completion of an image based on tiles finished. This is expected by Pixar's Alfred render queue, and a few other production queues.

ALF_PROGRESS ...%

VEX Profiling prints out information on shader computation at the completion of the render. This will help identify bottlenecks in the shading process, but will extend render times, so it should only be used for testing. There are two setting for this execution profiling and profiling and NAN detection. When NAN detection is turned on, each instruction executed in VEX will be checked for invalid arithmetic operations. This will check for division by 0, numeric overflow, and invalid operations. Errors like this will typically result in white or black pixels in the resulting image.

Example Listed on Following Page

Verbose Level is a 0 to 5 scale level of the amount of statistics to be produced by the renderer. 1 to 2 is usually a good setting. 5 is pretty unreadable.

Level 1

Generating Image: ip (720x405) // Line 1 - Begin

Plane[C]: Cf+Af[4] (16-bit float) // Line 2 - End of Header

Render Time: 1:46.611u 0.265s 1:49.99r // Line 3 - Begining of Footer

Memory: 516.55 MB. VM Size: 705.69 MB // Line 4 - End

Line 1 - is the render Directory and size of the image.

Line 2 - is the list of image planes and bit depth.

Line 3 - u is the amount of user time mantra took to render the image. This might not be 100% accurate since multiple cores or threads can compound this, for instance Linux, plus different OS have different system variable definitions. s is the system overhead time to read, or write the file, etc. r is the total time it took to render. The equivalent to this is in the Render Scheduler.

Line 4 - Arena Size is the amount of memory mantra allocated for the render, not what was actually used. VM, Virtual Memory, is the amount of memory reported by the OS, and may significantly exceed what is actually used.

Mantra needs to grab continuous chunks of memory as it builds data structures. Once it frees up the data, the OS controls the arena size shrinking it where it finds continuous chunks of memory back to the free pool of available memory. This is called memory allocation and memory deallocation. You don't want the arena size much larger than the actual memory used.

Level 2

// Parameters added after the Level 1 Header

SampleFilter: alpha

PixelFilter: gaussian 2 2

```
   VEX Type:     vector4
   Dither:       0.5
   Gamma:        1
   Gain:         1
   White point:  1
Load Time: 0.655u 0.93s 0.64r
Memory:  81.68 MB. VM Size: 324.31 MB

Level 3 -
// Shader calls adds after the Level 2 header
VEX Shaders Loaded
  op:/shop/mantrasurface stdin.vex
  op:/shop/mantrasurface stdin.vex
Thread Count: 8

//Displacements calls added prior to the Level 1 footer
Object /obj/Shader_Tester_Global1/Art displaced 5.17417e-011 (bounds set to 0.05)
Object /obj/Shader_Tester_Global1/Ground displaced 7.43967e-010 (bounds set to 0)
//Geometry count added after the Level 1 footer
Peak Geometry Objects: 7

Level 4 - nothing more with a basic scene

Level 5
//Bucket calls added after the Level 3 header
Render: X(415,430) Y(399,405)
```

Will add this nearly endlessly as it list bucket locations. It will over flow your terminal window and you will loose the more important information.

There are many other calls that can be generated at the higher levels, like oct-tree building, shader errors, instance loading, python scripts running, and many more.

A Typical VEX profile call

The invocations instruction has been put into collums to fit on this page. This is the call from the Mantra Surface Shader. The function calls of each shader are intended to be read as collumns.

VEX profile listing generated: Mon Sep 20 19:58:31 2010

Function	Excl. Secs	Incl. Secs	Calls	Instructions	Instr.	Local Per Call	Storage
pathtracer	719.58	835.29	2,357,985	40,242,132,435	17066.32	1,193,068	
opdef:/Shop/v_asadlight	0.-70	0.04	663,352	7,960,224	12.00	57,480	
op:/shop/mantrasurface	-11.-19	0.50	1,518,568	21,259,952	14.00	110,712	
op:/shop/mantrasurface	-107.-6	1.44	2,357,985	223,345,223	94.71	631,780	
opdef:/Shop/v_arealight	-226.-97	17.25	193,200,520	6,182,416,640	32.00	24,896	
opdef:/Shop/v_rayshadow	-373.-57	184.06	82,490,792	1,512,352,063	18.33	118,960	

Invocations	Instruction
15,118,929,285	set
4,310,059,565	if
4,302,732,785	endif
3,472,327,264	mul
3,110,554,956	add
2,151,168,754	div
1,764,008,013	gt
1,760,679,953	and
1,558,877,197	luminance
968,380,247	band
872,582,307	not
778,879,008	lt
652,824,764	sub
649,495,387	renderstate
591,107,638	else
586,900,501	eq
577,524,612	neg
465,587,615	length
419,858,270	while
398,351,046	max
393,689,717	jump
391,092,233	ge
384,463,464	nextsample
378,696,443	variance
375,864,365	sample_bsdf
288,355,795	or
195,709,691	eval_bsdf
193,600,254	sample_light
188,592,856	intersect_lights
161,472,573	simport
122,507,218	shadow_light
95,865,471	normalize
79,559,472	lerp
79,486,920	trace
43,093,561	getcomp
26,598,244	bwhile
20,183,588	limport
11,781,827	sqrt
10,091,761	setcurrentlight
9,431,940	setcomp
9,431,599	albedo
4,715,970	arraylength
4,715,970	bnot
4,715,970	bor
4,715,970	dot
4,715,970	min
4,715,970	popinline
4,715,970	pushinline
3,037,136	ptransform
2,357,985	diffuse
2,357,985	fresnel
2,357,985	getbounces
2,357,985	getlights
2,357,985	le
2,357,985	normal_bsdf
2,357,985	phonglobe
2,357,985	switch
1,518,568	computenormal